CLARKSON UNIVERSITY

Detection of Swipe Pressure using a Thermal Camera and ConvNets for Natural Surface Interaction

Honors Thesis by

Timothy J. Dunn

Department of Computer Science
Department of Electrical Engineering

Clarkson Honors

May 2019

Accepted by the Honors Program

Advisor (Sean Banerjee)

Advisor (Natasha Banerjee)

Honors Reader (Reader Name)

Honors Director (Jon Goss)

Abstract

In this paper, I present a system for reliably distinguishing between two levels of applied finger pressure on planar surfaces using a thermal camera. This work is the first to do so without requiring prior per-user calibration, and will enable arbitrary natural materials to be used as touchscreen surfaces in augmented reality applications. Two approaches were explored during this research for swipe pressure identification, which took place over the Spring and Fall Semesters of 2018. The first approach used morphological filters and supplied handcrafted features as input to a random forest classifier. The second approach used convolutional neural networks to classify both raw and filtered video data using several approaches. It was found that convolutional neural networks could only consistently outperform the random forest classifier when the same morphological filtering had been applied on the input videos.

Contents

List of Figures

1 Introduction

1.1 Problem Statement

Modern LCD touchscreens are able to recognize a wide variety of user interactions such as tapping and swiping, each of which elicit a unique software response. When working with projected images (e.g. projecting a screen onto an ordinary wooden desk to create a "smart desk"), this type of interaction is not possible due to the lack of touch sensors embedded into the natural surface. In order to demonstrate that it is feasible to remotely sense subtle differences in user swipe gestures, I have developed a system which can distinguish between low and high pressure finger swipes without requiring per-user calibration beforehand. This particular differentiation was chosen due the difficulty of the task (with an RGB camera, the pressure difference is indistinguishable due to identical fingertip motion) and a widespread lack of touchscreen support for this feature. Successfully and reliably detecting finger swipe pressure demonstrates that thermal cameras can be used to recognize and analyze user gestures on arbitrary natural surfaces. This greatly improves the adaptability of touchscreens, enabling them to be used where installing touch sensors in unfeasible or impractical.

1.2 Objectives

The objectives of this research were to accurately detect the pressure difference between hard and soft swipes on a natural planar surface using a thermal camera, without requiring prior per-user calibration. Swipes were performed and recorded on wooden, laminated, drywall, cloth, concrete, and whiteboard surfaces, and classifiers were trained independently for each material. Each classifier was trained using a leave-one-out approach, where it learns from other users the patterns which correspond to soft and hard swipes, and then attempts to properly classify the swipes of a newly introduced user. This represents a realistic scenario for a commercially deployed system, and it is highly desirable to avoid requiring per-user calibration for hard and soft swipes to improve classification accuracy.

1.3 Limitations

Several simplifications have been made to our testing environment which cannot be expected in an application setting. Firstly, users were instructed to wait several seconds between performing each swipe to allow the heat from the previous swipe to fully dissipate from the surface before recording the next one. In order to account for this, in future work a more sophisticated method of background subtraction must be developed to account for regions where a swipe has been most recently performed. Secondly, because each video in this study contains a single swipe, our classifier will not be effective in a multi-user environment where multiple actions may occur simultaneously.

1.4 Broader Impacts

As augmented reality systems grow in popularity and sophistication, people will increasingly interact with the real world through these partially immersive devices. Currently, users are limited to interacting with virtual environments through handheld controllers and arm gestures. As technology improves, reliance on controllers will decrease and users will be able to interact directly with their surroundings. In this scenario, thermal cameras could be incredibly useful for effectively capturing and quantifying a user's interaction with their environment, such as selecting items on a planar menu or grasping objects.

2 Literature Review

2.1 Thermal Imaging

On the electromagnetic spectrum, thermal radiation is classified as infrared light (light with a wavelength of 800nm to 1mm), since it has a wavelength of 7 to 15μm [7]. At this frequency, electromagnetic waves are not visible to the naked eye and must be detected using a sensor called a microbolometer, shown in Figure 2.1a. As thermal radiation strikes a microbolometer, it absorbs heat and becomes less electrically resistive. By measuring the minute changes in resistivity of a microbolometer pixel array (see Figure 2.1b), an image can be generated which visually depicts the relative quantities of heat radiated from an object. The quality of microbolometers has increased substantially in recent years, and high-quality thermal cameras can now capture images with 640×480 pixel resolution [8].

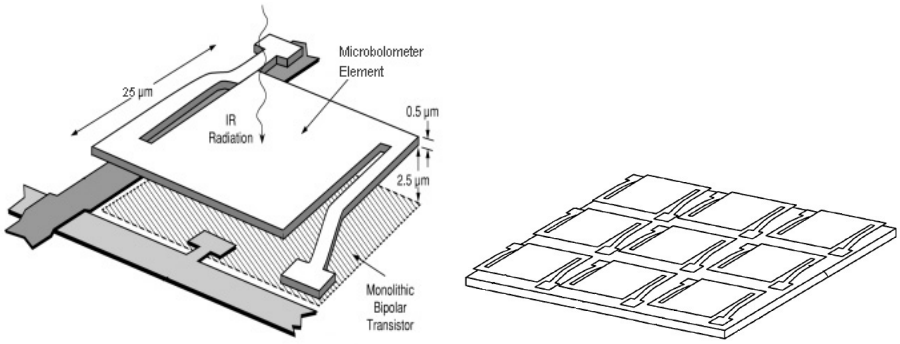

Figure 2.1: (a) Single Microbolometer (b) Microbolometer Pixel Array

2.2 Finger Gesture Tracking

Work in this field began in 1993, when Pierre Wellner designed the "DigitalDesk", an extension of the ordinary workplace desktop which included projected content [10]. A schematic of the DigitalDesk is shown in Figure 2.2. Wellner tracked user's finger movements by computing the difference of consecutive frames. This simple technique, although useful for identifying regions of a video where motion has recently occurred, is neither able to distinguish hand movement from other movements which may occur nor pinpoint exactly where the user's fingertip is located.

Oka et al. improved upon previous work in 2002 with the revolutionary idea of using thermal cameras to identify user's fingertips [11]. Reasoning that infrared cameras would be impervious to changes in lighting, patterns in projected content, and complex or dynamic backgrounds (issues which had plagued similar research endeavors in the past), Oka et al. successfully performed real-time tracking of multiple fingertips using a thermal camera. Several previous approaches had relied on detecting the user's hand based on a global "skin color" (which is not particularly robust given the wide range of possible skin colors) or using colored markers (which are cumbersome and would not realistically be used for an actual application). First, the center of the user's hand is detected by thresholding the image and eroding the result (see Appendix A.3.1 for an explanation of morphological filtering) to identify the region of the user's hand farthest from a boundary. At this point, several parameters must be initialized specifically for each user, such as their approximate hand and finger sizes. Next, template-based matching is performed to identify all visible fingertips, correlations are determined between frames, and their trajectories over

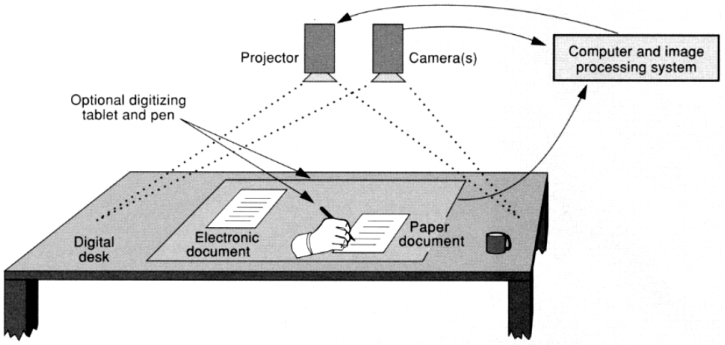

Figure 2.2: The DigitalDesk [10]

time are mathematically modelled. The researchers could distinguish between a circle, square, and triangle being drawn on the desktop surface with 99.2% accuracy. Figure 2.3 shows three extended fingertips being tracked in real time as a circle is drawn.

Figure 2.3: Real-Time Fingertip Tracking [11]

In 2005 Iwai and Sato developed the "ThermoTablet", a painting application which enabled temperature-based interactions with a painting canvas [12]. The back-projected canvas surface was digitally updated based on the temperature of the canvas, which was detected using an infrared camera also located behind the translucent canvas.

At the same time, Andrew Wilson developed a stereoscopic system which could detect a user's hand as it hovered above a touchscreen [13]. Unlike the passive measurement devices used by Oka et al., Wilson paired an infrared LED light source with a camera overlaid with an infrared band-pass filter and mounted them overhead. Thus, instead of passively detecting natural infrared radiation emitted from a user's hand, Wilson's system bathed the entire scene in infrared light, and then measured where the light was blocked by the user's hand. Based on the shape of the resulting shadow, Wilson could pinpoint not only the user's 2D fingertip location on the projected surface, but also the height at which the user's hand was hovering above the surface. Figure 2.4 shows Wilson's design in more detail.

The "SixthSense" wearable gestural interface developed by Mistry and Maes in 2009 is a pendant-like mobile wearable device comprised of a mini-projector, a mirror, and a camera [14]. The projector can display visual information on any nearby surfaces or physical objects while the camera is used to recognize and track the user's hand gestures. In [17], Kurz demonstrates that thermal cameras can be used to accurately identify contact points between a user's finger and an arbitrary object or surface. Together, these results foreshadow the future ubiquity and potential applications of wearable augmented reality devices.

In 2011, Larson et al. explored the detection of shape-based, pressure-based, and multi-finger gestures using thermal cameras [15]. Larson first denoised each frame using a moving-average filter, then thresholded using Otsu's algorithm [3], thinned the resulting image, and applied a hit-or-miss transform (see Appendix A.3.1) to the image with a rotating 3×3 structuring element to determine the endpoints of each finger. The search space is constrained by zeroing pixels which were not located within a hand segmentation in the past second, and the detection of heat

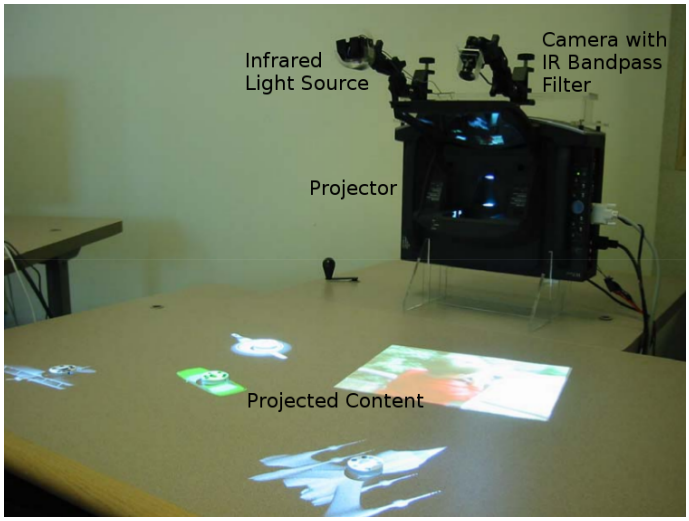

Figure 2.4: PlayAnywhere Finger Tracking System [13]

traces is solved as a Bayesian estimation problem. The parameters for this system are calibrated to each user and surface, by allowing them to first swipe at multiple pressure levels.

Saba et al. combine depth sensing using a Microsoft Kinect and temperature sensing using a thermal camera in order to detect surface touch points and classify the pressure of chording motions [16]. They first perform static background subtraction, threshold each frame using Otsu's algorithm [3], and then perform an opening operation on the resulting binary image to remove any gaps in the segmentation (see Appendix A.3.1). Each connected component is processed using the Convex Hull method, and the resulting vertices are assumed to be the fingertips. To reduce false positives, a distance constraint between the centroid of the hand and each fingertip is enforced, and a hit-or-miss transform using a disk-shaped kernel is performed around each point on the convex hull. Residual heat is detected as in [15]. Although the authors claim to be able to distinguish between various pressure levels, no results are included in the paper.

2.3 Convolutional Neural Networks

Convolutional neural networks are widely used to perform image classification, and the second phase of my research depended heavily upon their usage (please read Appendix A.2 for an introduction). CNNs were first introduced in 1999, by LeCun et al. [19]. This groundbreaking work replaced previous object and shape recognition approaches which involved a pipeline using image segmentation, feature extraction, and object classification using template matching, discriminative models, or generative methods. Instead, gradient-based learning was applied to exploit strong local 2D image structure and reduce the number of model parameters required by fully-connected artificial neural networks (see Appendix A.1 for an introduction). This foundational paper classified grayscale 32×32 pixel images from the MNIST dataset of handwritten digits [4] with 99.2% accuracy [19]. This initial network's architecture consisted of two repetitions of convolution and subsampling, followed by two fully-connected layers, shown in Figure 2.5. Improvements over this initial architecture such as max-pooling were introduced in later years.

In 2013, 3D CNNs were first developed by Ji et al. in an attempt to incorporate temporal information into a video classifier which could perform human action recognition [20]. From each input frame, they generated five channels corresponding to the grayscale, x-gradient, y-gradient, x-optical flow, and y-optical flow representations of the image. Their network architecture was similar to [19] in that it incorporated several repetitions of convolution and subsampling followed by a fully-connected classifier, but differed in the fact that each convolution used a three-dimensional kernel which additionally incorporated information from adjacent frames. A diagram contrasting 2D and 3D convolution is shown in Figure 2.6, where the temporal dimension shows consecutive video frames.

Karpathy et al. in 2014 investigated multiple approaches for fusing per-frame information contained in videos

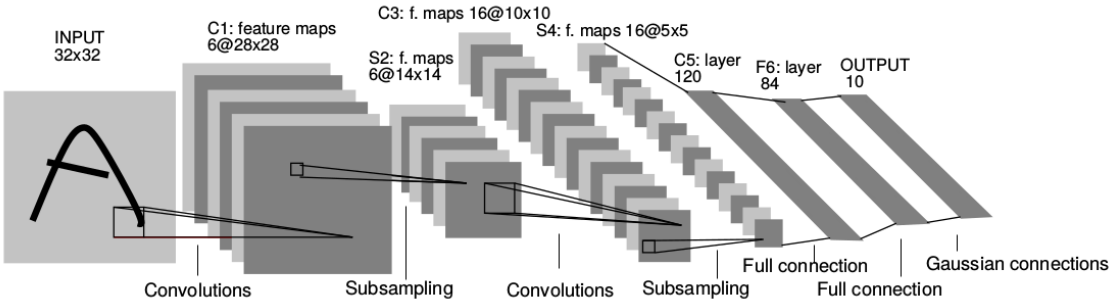

Figure 2.5: First Convolutional Neural Network Architecture [19]

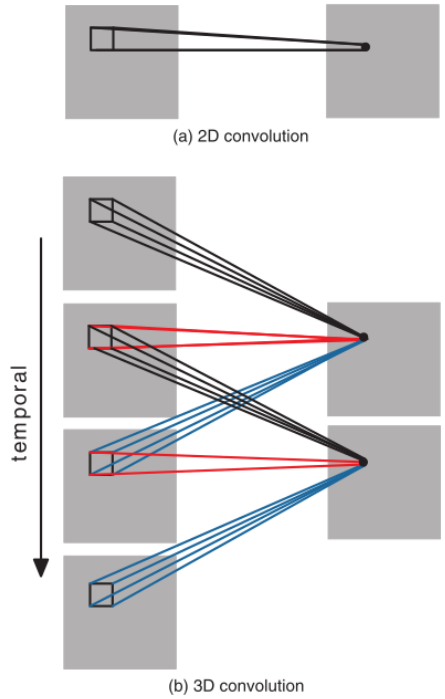

Figure 2.6: Three Dimensional Convolution [20]

during a large-scale video classification study [21]. They evaluated CNNs on 1 million YouTube videos to classify each into one of 487 actions, and investigated Early, Late, and Slow Fusion approaches. These methods are shown in Figure 2.7, where red, green, and blue boxes indicate convolutional, normalization, and pooling layers, respectively. Please note that Early Fusion is equivalent to the 3D convolution used by Ji et al. in [20]. Late Fusion places two separate single-frame networks (with shared parameters) several frames apart, and then merges the two streams in the first fully-connected layer. Slow Fusion is a balanced mix which gradually expands the scope of both spatial and temporal information used by the network. Although these spatio-temporal networks performed best, they gave a surprisingly small improvement over single-frame models (63.3% versus 59.3% accuracy on the UCF-101 dataset, which consists of $13,320$ videos of 101 human action categories [5]).

Later that year, Simonyan and Zisserman showed that using a two-stream CNN network with the original video frames and dense optical flow representation could achieve a significant improvement over Slow Fusion, which until then had performed best. They found that although performing 10-frame slow fusion achieves 56.4% accuracy on the UCF-101 [5] dataset (which is an improvement over single-frame classification accuracy of 52.3%), single-frame and 10-frame optical flow performed much better, with classification accuracies of 73.9% and 81.0%, respectively. By pre-training their classifiers and fusing both approaches using a Support Vector Machine (SVM), Simonyan and Zisserman achieved 87.0% classification accuracy on the UCF-101 dataset [22].

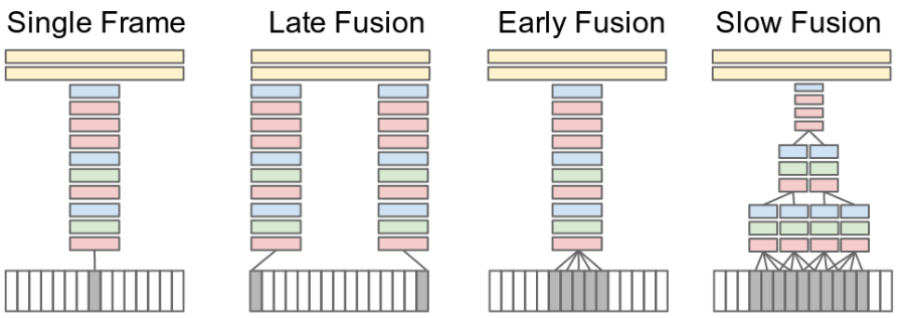

Figure 2.7: Various Frame Fusion Approaches [21]

In 2015, work by Ng et al. [24] showed that Long Short-Term Memory cells could be incorporated into current ConvNet architectures to more successfully aggregate information learned over longer periods of time and perform accurate classification. Later work by Varol et al. [30] avoided LSTM and instead used long-term convolutions. When used in a two-stream network with RGB and optical flow information, a 91.8% classification accuracy was achieved on the UCF-101 dataset when iDT (improved Dense Trajectories) information was available.

Tran et al. [23] investigated numerous ConvNet architectures and found that a homogeneous structure of $2 \times 2 \times 2$ pooling kernels and $3 \times 3 \times 3$ convolution kernels performed best, achieving 85.2% classification accuracy on the UCF-101 dataset. When optical flow or iDT were incorporated, this accuracy rose to 90.4%. Figure 2.8 presents an overview of the advances in ConvNet architecture for video recognition outlined above.

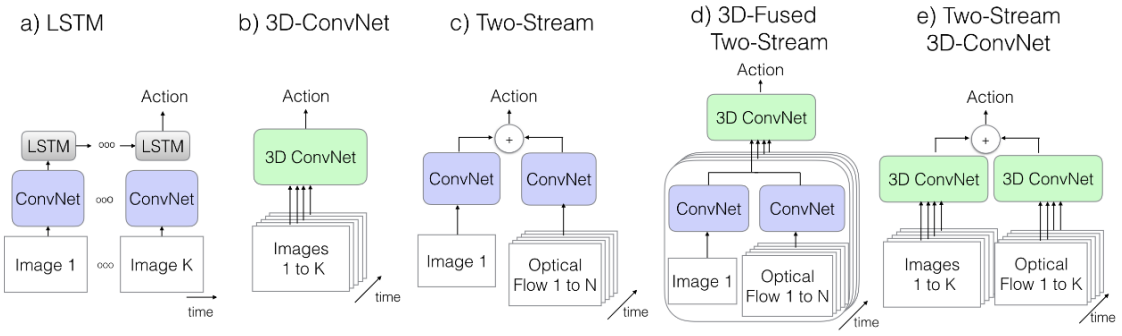

Figure 2.8: Progression of ConvNet Video Classification Architectures [28]

In 2017, Jiang et al. [27] were the first to apply recent advances in ConvNet action recognition to the infrared spectrum, using the InfAR dataset, created in 2016 [6]. Additional work in investigating ConvNet architectures was performed by Diba et al. in [29] and by Feichtenhofer et al. in [26], who found that ConvNet stream fusion is best performed in later convolutional layers.

3 Methodology

3.1 User-Independent Swipe Pressure Classification

The initial study occurred during the spring semester of 2018, including data collection, analysis, and classification. The purpose of this study was to demonstrate that thermal cameras can be used to remotely distinguish between hard and soft finger swipe pressures on natural surfaces without requiring user-specific information.

3.1.1 Capture Setup and Data Collection

A FLIR Vue Pro 640 × 512 30fps thermal camera was mounted overhead on a tripod facing downwards towards the surface of interest. The two surfaces chosen for the purposes of this study were paper and plywood, due to their low thermal reflectances and relatively high emittances. Video was recorded in 8-bit MJPEG format using a linear white-hot temperature encoding scheme. The approximate temperature range selected on FLIR's Android application was from $-25\,^{\circ}\mathrm{C}$ to $135\,^{\circ}\mathrm{C}$.

Nine test subjects performed 32 swipes of any length or direction onto the test surface, as if they were interacting with a large touchscreen. The test subjects were both male and female, all right handed, and ranging in age from 18 to 37 years old. The first 16 swipes were soft (low pressure) swipes, and the remaining 16 swipes were hard (high pressure) swipes. Swipes were performed at regular intervals, with a 5 second delay between each soft swipe and a 10 second delay between each hard swipe. This allowed for heat to fully dissipate from the surface each time before the next swipe was performed. Data was recorded as a single video and later segmented into 32 clips, each containing a single swipe. An example video frame is shown in Figure 3.1.

Figure 3.1: Recorded Thermal Finger Swipe

One issue which was consistently encountered during data collection is that FLIR Vue Pro thermal cameras will dynamically rescale the intensities of each pixel based on the minimum and maximum temperatures currently sensed within the image. This poses a problem because an object with a constant temperature will not be represented by a constant pixel intensity in consecutive image frames. Considerable effort was spent attempting to disable this

"feature", but eventually it was determined that both hardware triggers and software flags are ignored by the FLIR software. Upon conclusion of this study, all FLIR thermal cameras were returned and new cameras were purchased. As a temporary workaround, I placed my hand in the corner of each video as it was captured to provide a steady maximum temperature. The hand would eventually be cropped out by the video processing pipeline, described in the next section.

3.1.2 Swipe Path Extraction

Each video containing a single swipe was first cropped to a predefined region of interest, selected so that it would contain only the user's hand, swipe, and the constant background surface. Next, static background subtraction was performed and each video frame was binarized using a constant thermal intensity threshold of 4 on a scale of 0 to 255. A series of morphological filters with various kernel sizes was used to remove the user's hand and any remaining noise from the video (see Appendix A.3.1 for an explanation of morphological filters). First, a 3×3 kernel was used to apply opening and closing operations to the images and remove salt-and-pepper noise. Then, the image was dilated using an 11×11 kernel to fill in any narrow gaps present between a user's fingers. In an effort to isolate the swipe path, narrow lines were first removed by performing opening and closing operations on the image with kernels of size 25×1 and 1×25. Since the swipe path was removed by this operation, the difference between the resulting and previous images was calculated, which contained narrow lines such as the swipe path. A final opening operation was performed with a 7×7 kernel in order to remove noise remaining near the edge of the user's hand. Remaining white pixels were considered to be the region of the image containing the user's swipe, and then this result was applied as a per-frame mask over the original video. The intermediate results at each stage of this pipeline are shown in Figure 3.2.

Figure 3.2: Thermal image pipeline for swipe extraction

The video was then transformed into a single image by summing pixel intensities over time. Since the vast majority of user swipes were simple arcs, each swipe was approximated by a translated and rotated quadratic function. In order to do this, the line of best fit was first calculated and the coordinate system was transformed so that the line of best fit lay along the $+X$ axis. Afterwards, the quadratic of best fit was calculated and the "swipe path" was defined by pixels which when transformed lie within 10 pixels of this quadratic approximation. An example polynomial swipe path approximation is shown in Figure 3.3, laid over the summed-pixel image.

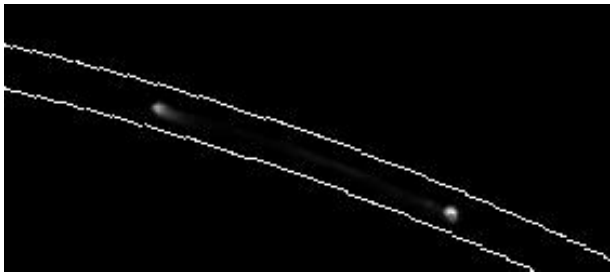

Figure 3.3: Polynomial Swipe Approximation

3.1.3 Swipe Classification

Using only the pixels which were contained within the swipe path, 30 features were calculated. The first 10 features were the average pixel intensities during each 1/2 second time interval of the first 5 seconds of each swipe. The last 20 features were the maximum and average pixel intensities within each bin of a 10-bin histogram along the swipe length in the accumulated intensity image. A 500-tree random forest classifier was trained on these 30 features, with each split in the tree using at most 6 features. To perform user-agnostic classification, I performed leave-one-out cross-validation using all 16 hard and soft swipes from the 8 remaining users. In order to determine the impact that incorporating user-specific information would have on classification accuracy, I then re-trained the classifier while incorporating a gradually increasing number of samples from the user whose swipes were being classified.

3.2 Towards More Robust Swipe Pressure Classification

The second study occurred during the fall semester of 2018, and was performed as an extension of the original study which explored the utility of ConvNets in swipe pressure classification, and expanded the set of material surfaces used. All of the source code for this study is available on Github, at `https://github.com/TimD1/HonorsThesis`. Using existing platforms such as Keras, Jupyter Notebook, and Docker greatly facilitated data analysis and classification.

3.2.1 Capture Setup and Data Collection

A FLIR Vue Pro 640×512 30fps thermal camera was again used for data collection, and video was captured with the same encoding and parameters as in Section 3.1. To investigate the performance of the new classifier on a variety of surfaces, data was collected for user swipes on concrete, wood, laminant, cloth, plastic, and drywall. Ten test subjects performed 25 soft swipes and 25 hard swipes for each material, again with a 5 second delay between soft swipes and a 10 second delay between hard swipes. This provided us with a significantly larger dataset than was collected during the previous semester.

3.2.2 Training Data Generation

During initial processing, a video segmentation script was used to segment one video recording into many clips, which each contained a single swipe. When segmenting each video, a 256×256 pixel region of interest was manually selected for each swipe. Selecting a uniform size for the region of interest was critical for this study because a convolutional neural network requires an input layer of fixed size. Figure 3.4 outlines the fixed-size region of interest selected for an example video.

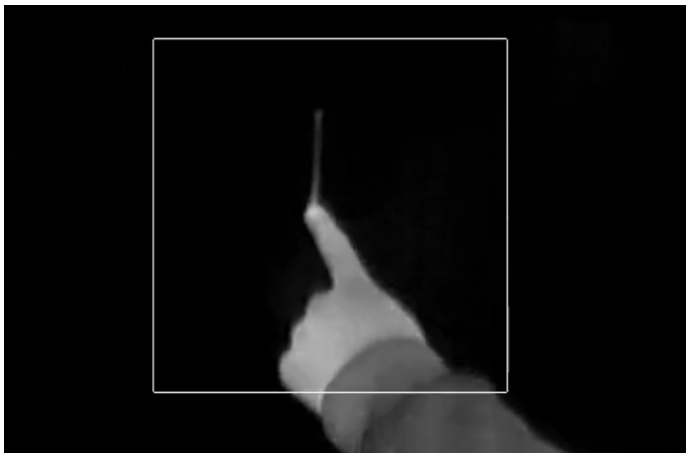

Figure 3.4: 256×256 Pixel Region of Interest

It would be undesirable to use every frame within a swipe-containing video to train a ConvNet classifier. Numerous frames near the beginning and end of each video neither coincide with the user's swiping action nor show

heat dissipating from the background surface. As such, including these frames would only introduce noise to the classifier and prevent it from learning useful information from the training data. To select only the most relevant frames from each video, a plot of average frame intensity over time was first generated to gain an intuition of how this value develops over time. Afterwards, several heuristics were tested to determine the optimal approach for extracting relevant frames. An example plot is shown below in Figure 3.5 which displays the average frame intensities over time for *user6*'s hard swipes on a wooden surface.

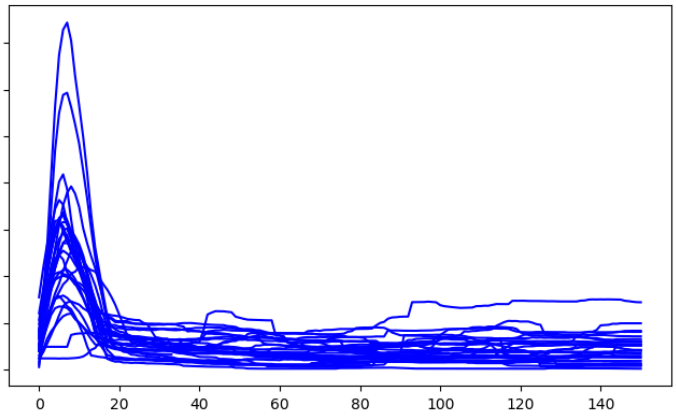

Figure 3.5: Average Frame Intensity Over Time

The most successful heuristic tested required finding the first and last frames where the average pixel intensity exceeded the average pixel intensity during the whole video. These two boundary points typically occurred when the user's hand entered and left the region of interest. It was noticed that the time intervals required for each user to place their hand on the surface, perform a swipe, and remove their hand were approximately equal. Thus, the first third of frames within this time interval was ignored (since the user had not begun performing the swipe) and the central 8 frames of the remaining two-thirds of the frames within the interval were stored for further analysis.

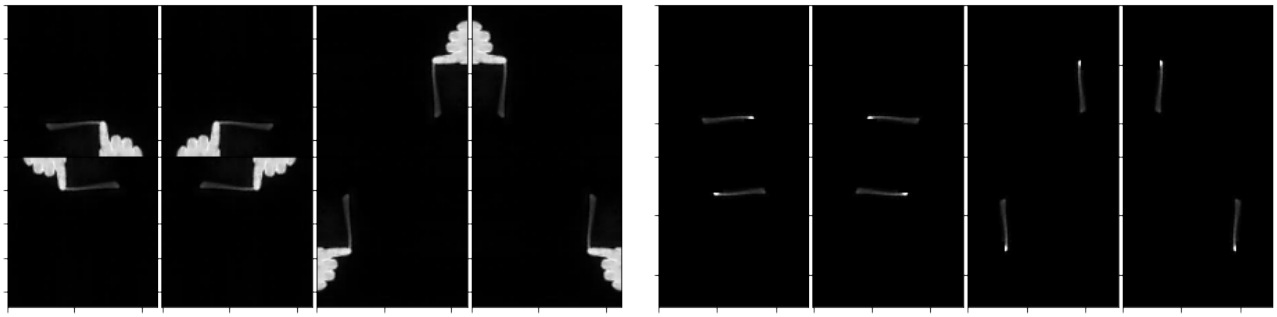

Figure 3.6: Data augmentation (a) with, and (b) without the user's hand

After these 8 frames were extracted from each video, data augmentation was performed to ensure that the resulting classifier was robust. Each video frame was rotated in intervals of 90 degrees and reflected once, resulting in an augmentation factor of $8x$. Additional operations were not considered in the interest of time since there were no Python libraries readily available for performing video data augmentation.

3.2.3 ConvNet Architecture Design

The reader may find it useful to first read Appendices A.1 and A.2. The first architecture used for swipe pressure classification was modeled after VGGNet [25], and was composed of five repetitions of 3×3 convolution, 2×2 max-pooling, and 0.25 dropout layers followed by flattening and a single fully-connected layer to perform the final binary classification. Input pixel intensities were rescaled to lie on the interval $[0, 1]$. This architecture also closely resembled that used by Tran et al. in [23]. Unfortunately, it quickly became clear that insufficient training data

was available to train such a deep neural network from scratch. Figure 3.7 shows the results of an early attempt at CNN training where the network was unable to learn useful information and did worse-than-random classification on the validation data.

```
Train on 23910 samples, validate on 5978 samples
Epoch 1/50
23910/23910 [==============================] - 26s 1ms/step - loss: 0.6928
- acc: 0.5111 - val_loss: 0.6957 - val_acc: 0.4433
Epoch 2/50
23910/23910 [==============================] - 25s 1ms/step - loss: 0.6928
- acc: 0.5156 - val_loss: 0.6963 - val_acc: 0.4433
Epoch 3/50
23910/23910 [==============================] - 26s 1ms/step - loss: 0.6928
- acc: 0.5154 - val_loss: 0.6960 - val_acc: 0.4433
```

Figure 3.7: Failing to train a deep ConvNet

In order to perform classification with the quantity of training data available, it was clear that both the depth and size of the ConvNet used would have to be significantly reduced. First, I subsampled the 256×256 images to 128×128, 64×64, and 32×32 resolution and generated visuals of the results in order to determine the minimum resolution which could capture sufficient information about the swipe. Based on these results, which are shown in Figure 3.8, 128×128 pixel resolution was selected. The number of repeated convolution and pooling sequences was reduced to two, and the number of kernels used for convolution at each layer was a modest 32. In order to capture a greater quantity of spatial information in just two layers, kernel size was increased from 3×3 to 5×5. This drastically decreased the number of parameters which the ConvNet had to learn to perform classification, and the final architecture (minus the flattening layer after the final pooling operation) is shown in Figure 3.9.

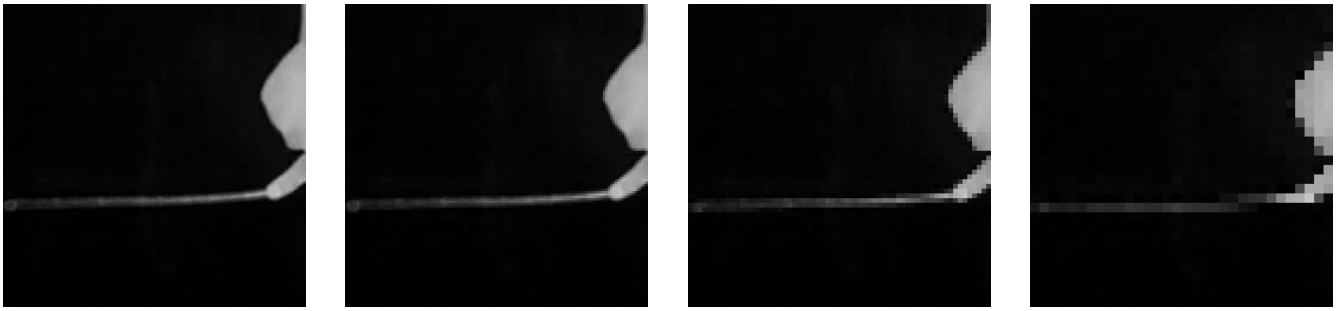

Figure 3.8: (a) 256×256, (b) 128×128, (c) 64×64, and (d) 32×32 image resolutions

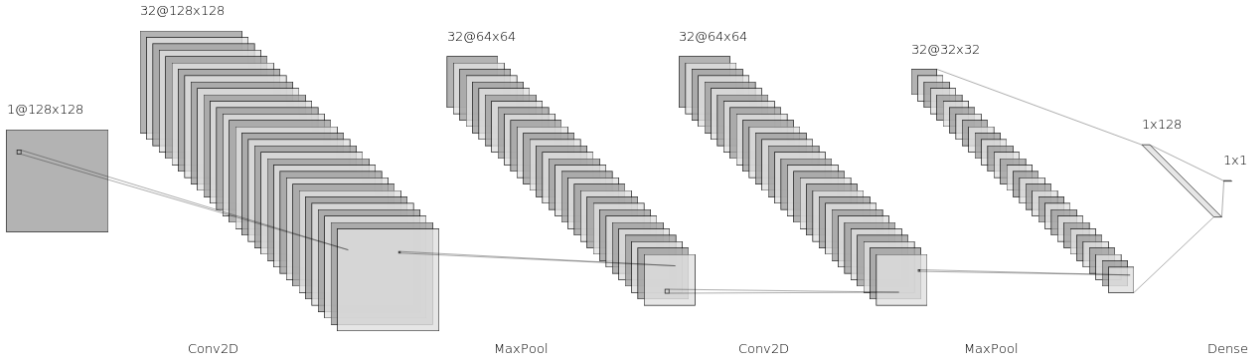

Figure 3.9: Simple ConvNet Architecture

3.2.4 ConvNet Extensions

Pre-training on VGGNet

In order to more successfully classify thermal swipes with limited training data, each image was first modified to be identical in format to images expected by the VGGNet architecture used for ImageNet classification. The initial layers of VGGNet would be borrowed with their parameters unmodified so that they could recognize basic visual patterns, and only the final fully-connected layers of the network would be retrained to classify swipe intensities instead of performing object recognition. Each training image was first resized to dimensions 224×224 pixels, the grayscale image was converted to its 3-channel RGB equivalent, and the mean values of each channel were subtracted as in the original paper [25]. Although the modified VGGNet classifier had no trouble correctly classifying the training data, it was prone to severe overfitting, as Figure 3.10 shows. This means that the classifier was learning to distinguish between the hard and soft swipes in the training dataset using methods which didn't generalize well. In order to rectify this, an aggressive dropout layer of 0.9 (90% weight dropout) was added before the final ConvNet classification layer, which alleviated the issue.

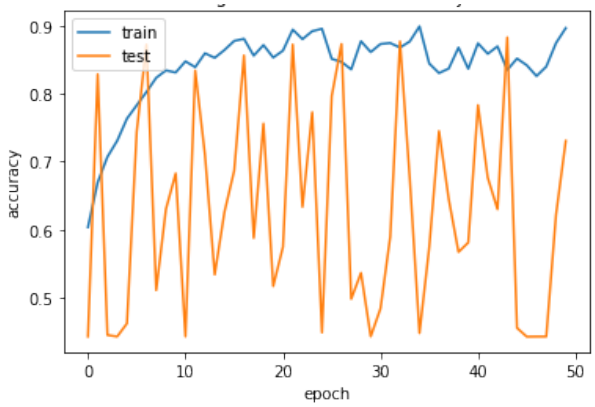

Figure 3.10: Overfitting with VGGNet Pretraining

Late Fusion: Averaging

In the simple ConvNet architecture described above and shown in Figure 3.9, the eight consecutive frames extracted from each video were classified independently. Intuitively, it should be possible to combine the information available in these eight consecutive frames to perform more reliable swipe pressure classification. Figure 3.11 shows eight consecutive frames with and without the user's hand. In order to perform late fusion (see Figure 2.7) with this information, both a simple averaging technique and a support-vector machine classifier were trained to merge the classification results from each of the eight video frames into a single video classification result. This step was in general not as useful as expected, and only the simple classification result averaging method reliably outperformed single-frame classification.

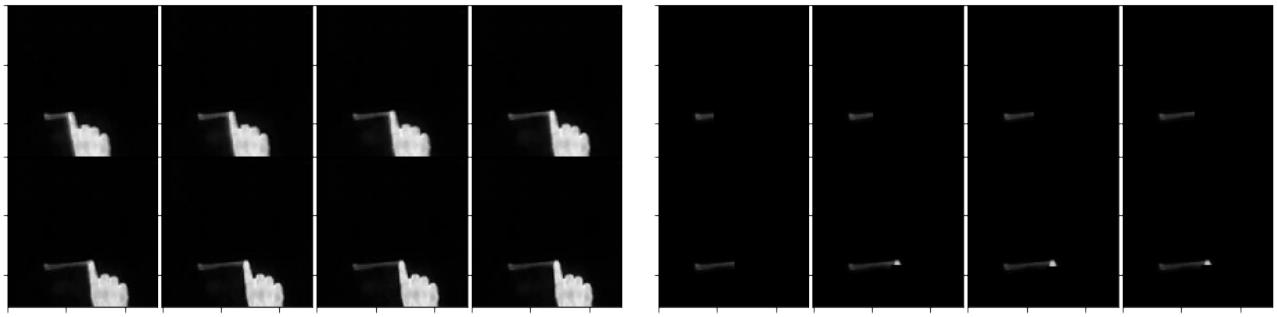

Figure 3.11: Eight consecutive video frames during swipe (a) with, and (b) without the user's hand

Early Fusion: Pixel Summation

In an attempt to merge per-frame information earlier in the pipeline, pixel intensity values were summed over time to create an image representing the total thermal intensity over the course of the video for each pixel. Naturally, this was done only for the video with the user's hand removed, and the VGGNet-pretrained classifier was used since it had shown the best performance thus far. Each resulting image is similar to that shown in Figure 3.3 but without calculating a polynomial swipe approximation. This method worked well, most likely due to the fact that averaging over many frames has the power to reduce the quantity of noise present if only a single frame were selected.

3D Convolution

Using three-dimensional convolution was investigated for this particular application, but with no positive results. Like the early ConvNet which was too deep to learn anything from the training data, a ConvNet with three-dimensional convolutions was too complex to extract useful information from the images provided. The most likely cause is that although the noise present in consecutive frames of a thermal video is different (which can be reduced by early or late fusion, as shown above), there are no significant temporal features which are useful for swipe pressure classification. The timing of user swipes appears to be invariant to pressure, as Figure 4.5b demonstrates. The only major useful piece of information which can be learned from temporal and not spatial information of these thermal videos is the rate at which the heat from a finger swipe fades from a material's surface, and is more a function of the material and the ambient room temperature than the swipe pressure. Solving a more complex classification problem without introducing additional information meant that three-dimensional ConvNets would not perform well in this application.

Optical Flow Calculation

Similar to three-dimensional CNNs, optical flow calculation is an incredibly neat and useful tool which is of limited application to classification of thermal finger swipe pressures. As Figure 3.12 shows, the optical flow vectors calculated for thermal finger swipe videos successfully identified the outline of the user's hand. Combination of optical flow information with raw video data is useful for action identification, where the region and characteristics of motion vary significantly. For this application, classification using optical flow would only detect subtle timing differences between hard and soft swipes. Given the low volume of training data available and lack of optical flow information concerning the thermal swipe, this avenue did not look particularly promising and was not explored further. Even in videos where the user's hand was removed, optical flow only detected the small artifacts from the user's fingertips which were not perfectly removed. This can be seen by comparing Figures 3.11b and 3.12b, and is understandable given that a swipe consists of fading intensity, rather than motion.

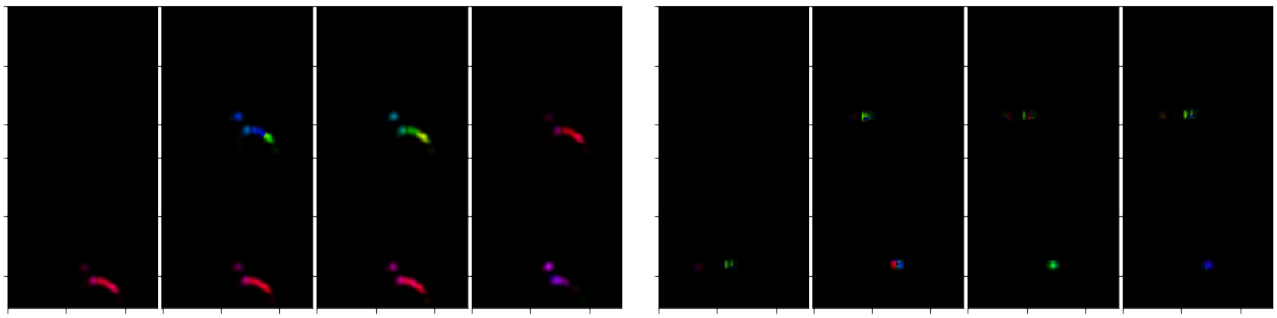

Figure 3.12: Optical flow calculations visualized (a) with, and (b) without the user's hand in the video

3.2.5 Directions for Future Work

Increasing the volume of training data, either through additional augmentation or data collection, represents the logical next step for this research. Although using a VGGNet-pretrained classifier on the summed-pixel image for each video had the best performance, skipping the data augmentation step resulted in per-material classification accuracies which varied each time the classifier was trained from 50% to 90%, instead of consistently achieving above 80% classification accuracy. Logical modifications in training data generation would involve rotating images in smaller increments, whitening, modifications of scale, image translation, and the intentional introduction of noise.

Continued effort towards improving ConvNet architecture is promising, since I made no effort towards modifying the learning rate and didn't adjust model architecture once classification was working sufficiently well.

Improved methods for swipe identification and hand removal would additionally be helpful. Artifacts such as the user's fingertips were sometimes evident in the processed video frames, and the combination of more complex hand modeling and identifying pixels which the hand has touched should increase the quality of image preprocessing, decrease the noise content of resulting processed frames, and improve the resulting classification accuracy.

A final long-term avenue for improvement is the development of machine learning techniques and models. ConvNets are most commonly applied to object recognition in images, where information is dense. For this particular application, the information content of the thermal images used is considerably more sparse, and less than 25% of each image's pixels are non-zero. Some modification of ConvNet behavior based on data sparsity could be useful.

4 Results

4.1 Characteristics of Thermal Swipes

4.1.1 Heat Signature

As can be seen below in Figures 4.2a and 4.2c, average pixel intensity was consistently higher for hard swipes. Hard swipes did have greater variance in intensity, though, possibly because users are less accustomed to performing them regularly in everyday applications. It can also be seen clearly in Figures 4.2a and 4.2b that in the average case, more heat was transferred to the desktop surface near the beginning and end of each swipe than the center. This is because most users pause slightly when starting and ending their swiping motions. In Figure 4.2c, average pixel intensity increases during the first three bins, after which it slowly decays. This shows that the average length of time which a user takes to perform a swipe is 1.0 to 1.5 seconds, after which no more heat is transferred to the material surface and the heat signature gradually decays.

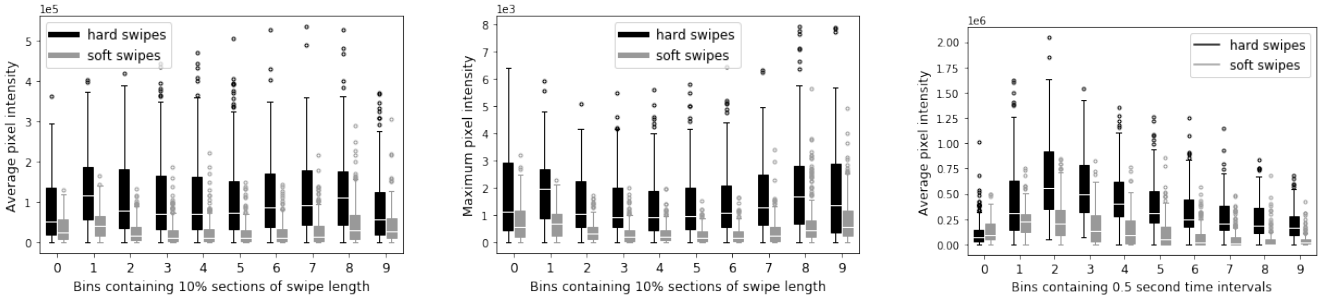

Figure 4.1: Boxplots of all features used for classification: (a) average pixel intensity binned along swipe length, (b) maximum pixel intensity binned along swipe length, and (c) average pixel intensity binned over time

After viewing several thermal videos, it was apparent that certain users started data collection with cold hands, most likely because they had walked over to participate in the study from another academic building and just returned from outdoors. For these users, their index fingers began cold but soon became considerably warmer than their average skin temperature after performing numerous hard swipes on the first material's surface. This observation is outlined in Figure 4.1.

4.1.2 Materials

From Figures 4.3a and 4.3b it is apparent that the average measured pixel intensity was greater for paper than for wood (please note the difference in scales). Despite this, the final classifier for the first study was better at recognizing swipe pressures on a wooden surface. Although the paper reached a greater maximum temperature than wood, it didn't require significant finger pressure to do so, and there was a less noticeable difference between hard and soft swipes on paper.

Material properties and initial temperatures impacted classification accuracy during this study, since not all materials reacted similarly to applied finger pressure. While materials such as wood and paper heated up fairly readily, the same was not true for the concrete surface. This is most likely because the concrete used in this study was a section of wall, similar to what would be used in a realistic application scenario. Since the concrete was anchored

Figure 4.2: Finger temperature at the (a) beginning, (b) middle, and (c) end of data collection for the first material

to the building's foundation, it was most likely a few degrees cooler than other materials. The whiteboard surface was unique in its reflectance of heat; when a user stood in the wrong place, their outline was clearly visible in the background. This was not avoided for the purposes of this study to simulate a real-world scenario and determine if the classifier could nevertheless determine the swipe pressure. As Figure 4.2.2 shows, the only classifier which had difficulties with this was the one which used an image of each swipe video summed temporally, magnifying this noise. Example images of swipes on select materials are shown in Figure 4.4.

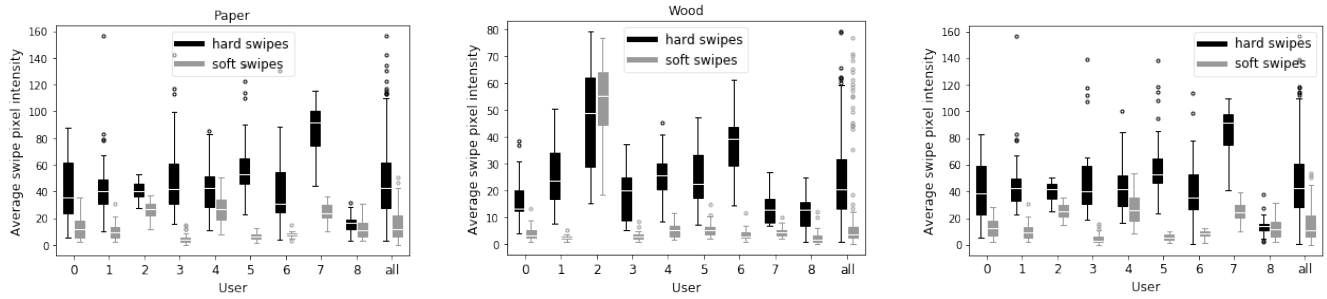

Figure 4.3: Per-user average swipe pixel intensities for (a) paper (b) wood, and (c) both materials

Figure 4.4: Example user swipes on (a) concrete (b) wood, and (c) whiteboard surfaces

4.1.3 Users

Figure 4.3 shows that users clearly have different opinions regarding what constitutes a "soft" or a "hard" swipe. For example, although *user7*'s hard and soft swipes were easily distinguishable from one another, with nearly no overlap in the range of average pixel intensities for hard and soft swipes, *user7*'s soft swipes consistently resulted in greater average pixel intensities than *user8*'s hard swipes. Clearly, simple thresholding is insufficient for accurate swipe classification, and it will be shown in Section 4.2 that allowing our model to incorporate user-specific information will positively impact classification accuracy. Each user's average swipe intensity profile can be seen in Figure 4.5 over both swipe length and time.

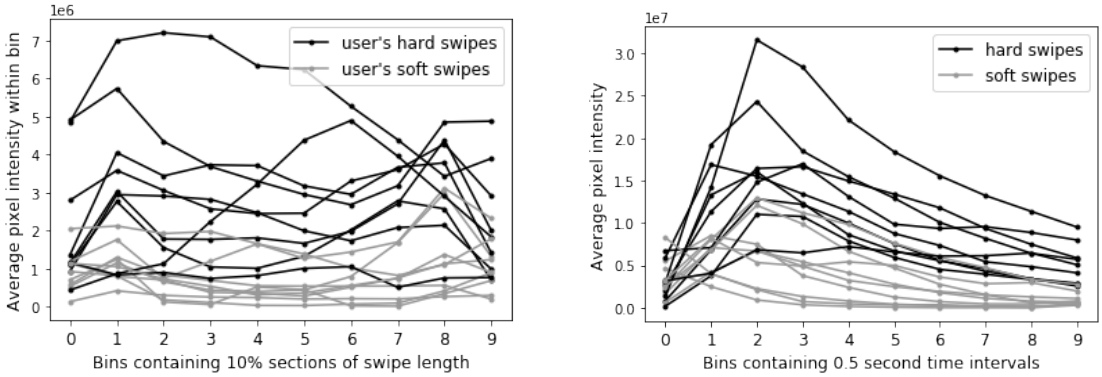

Figure 4.5: Per-user average swipe pixel intensities binned over (a) swipe length (b) time

4.2 Classification Accuracy

4.2.1 Random Forest Classifier

As described in more detail in Section 3.1.3, random forest classifiers were trained on the paper dataset, the wood dataset, and then both datasets combined. The resulting confusion matrices are shown in Figure 4.6, which contains the sum of performing leave-one-out classification for each user in turn.

Predicted Class

Actual Class	Soft	Hard
Soft	0.71	0.29
Hard	0.18	0.82

Predicted Class

Actual Class	Soft	Hard
Soft	0.93	0.07
Hard	0.16	0.84

Predicted Class

Actual Class	Soft	Hard
Soft	0.82	0.18
Hard	0.30	0.70

Figure 4.6: Confusion matrices for user-agnostic classification using (a) paper, (b) wood, and (c) both materials

After performing user-agnostic swipe pressure classification, I decided to investigate the impact that including example swipes from the user whose swipes were currently being classified would have on classification accuracy. Figure 4.7 shows the evolution of pressure classification accuracy for each user as an increasing number of training examples are provided from that user's data. It is interesting to note that the classifier had more difficulty distinguishing between pressure levels for some users than others, and its classification accuracy improved over time for some users and not others. The cause of this can be decided by more closely examining Figure 4.3. For example, the user-agnostic classification accuracy of *user7*'s swipes is around 65%. Moreover, introducing hard samples into the training dataset has no effect on classification accuracy, but once sufficient examples of *user7*'s soft swipe are provided, classification accuracy for *user7* quickly rises above 90%. This suggests that *user7*'s soft swipes are stronger than average, but that it is not difficult to distinguish between *user7*'s soft and hard swipes. The per-user histograms in Figure 4.3 confirm this hypothesis.

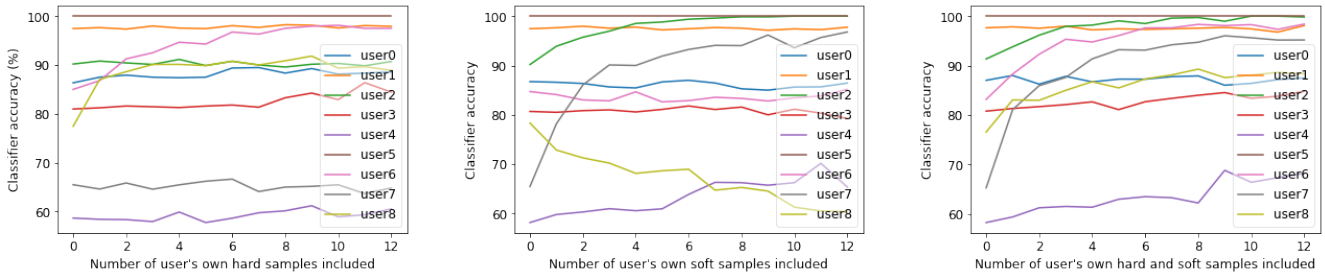

Figure 4.7: Leave-one-out classification accuracy over time as an increasing number of (a) hard (b) soft (c) hard and soft training examples from the user of interest are provided

4.2.2 Convolutional Neural Nets

Figure 4.2.2 shows the classification results for a single user's swipes after training ConvNets on the swipes of the nine remaining users, both by material and overall. My original intent was to perform 10-fold leave-one-out training and classification, and take the average classification rate to generate this table, but the number of approaches explored in this study prevented me from attaining a more generalizable result. The three classifiers which performed most successfully were the shallow ConvNet whose architecture is shown in Figure 3.9, VGGNet [25] with the fully-connected layer retrained on a single image, and VGGNet [25] with the fully-connected layer retrained on each video's summed pixel intensities.

		Ordinary CNN				CNN with VGGNet Pre-training				Sum CNN
		Hand		No Hand		Hand		No Hand		No Hand
		1-Frame	8-Frame	1-Frame	8-Frame	1-Frame	8-Frame	1-Frame	8-Frame	1-Frame
Material	Cloth	43.5	46.2	65.9	63.5	52.6	51.9	78.4	80.8	82.7
	Concrete	55.3	50.0	83.4	92.3	95.7	96.1	85.3	92.3	92.3
	Drywall	41.6	40.4	75.0	75.0	66.6	67.3	73.1	73.1	96.1
	Laminate	66.8	67.3	50.7	50.0	63.0	61.5	81.5	86.5	82.7
	Whiteboard	63.7	68.6	61.0	60.8	87.7	90.2	75.7	80.4	64.7
	Wood	53.1	53.8	73.1	80.8	54.3	53.8	71.4	78.8	94.2
	All	69.1	66.7	68.6	74.5	†	†	†	†	76.8

† These results were unable to be collected since all the training data could not fit on the GPU.

Figure 4.8: CNN Classification Results

Although the results certainly varied per material, it appears that using late fusion with 8-frame classification outperformed single image classification. Additionally, both pre-training on VGGNet and preprocessing by removing the user's hand appear to have improved classification accuracy. Overall, it appears that a classifier pretrained on VGGNet performed best at classifying an image with the hand removed and pixel intensities summed temporally. The one notable exception to this classifier's success is in performing classification on whiteboard swipes. This is most likely because the whiteboard is the only material with significant reflectance, and the consistent background heat emanating from the user's body becomes noticeable when each video is summed over time. Figure 4.2.2 affirms that VGGNet-pretrained classifiers had no difficulty classifying whiteboard swipes in general when a single image during the swipe is used. Combining the results of these two classifiers would most likely achieve the best classification accuracy; this was not attempted due to GPU memory constraints.

A Appendix

A.1 Artificial Neural Networks (ANNs)

A.1.1 Overview

Over the past decade or so, as computational power has significantly increased, neural networks have emerged as one of the most popular machine learning techniques. Modelled after biological neurons in the human brain, artificial neural networks are capable of approximating complex non-linear functions. Using advanced statistical methods, a neural network can be constructed which will with high probability map a set of inputs to the correct output, a process termed "classification". In order to generate such a network, where each node computes a weighted function of its inputs, the correct weights must be "learned" and the network "trained" by being provided with a large enough database of correctly mapped examples.

For example, if we wish to automatically classify grayscale images of handwritten digits (labelling each image with the identified digit) which are 28×28 pixels large, with the tools available today it is relatively straightforward to construct a neural network which does so with 90% accuracy by providing it with thousands of labelled examples (which are publicly available from the famous MNIST dataset [4]) and using various libraries to perform the numerical optimization. Based on these many correctly labelled examples, the neural network is capable of learning a function which maps all the pixel intensities in a new 28×28 image to an integer in the range 0 to 9. State-of-the-art neural network-based handwritten digit classifiers can now perform better than humans at the same task.

A.1.2 Artificial Neurons

The Perceptron

The most basic building block of a neural network is a single neuron, which like its biological analogue generates an output signal in response to various inputs. A direct visual comparison is shown in Figure A.1.

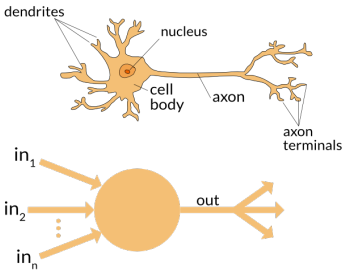

Figure A.1: Biological and Artificial Neurons [31]

We will begin our discussion of neurons with a perceptron, which is basically a simpler version of a neuron, where the output is simply a shifted step function applied to a linear combination of the input variables. This can be represented mathematically by the equation shown below:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq b \\ 1 & \text{if } \sum_j w_j x_j > b \end{cases} \tag{A.1}$$

where x is the vector of input variables, w is the vector containing the associated neuron weights, and b is a constant offset known as the "bias". A graph of a unit step function time-shifted by constant b is shown in Figure A.2.

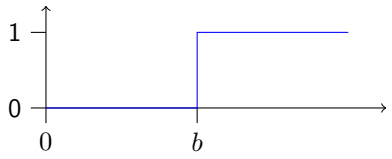

Figure A.2: Unit Step Function

An Example Application

For example, if we would like to use a single neuron to predict whether or not a freshman will pass his second chemistry test of the semester, we can use input variables $x = \langle x_1, x_2, x_3 \rangle$, weights $w = \langle w_1, w_2, w_3 \rangle$ and bias b:

$$\begin{aligned} x_1 &= \text{a student}'\text{s score from the first test} \\ x_2 &= \text{student}'\text{s average score on all homework assignments} \\ x_3 &= \text{difference in hours spent studying for the two exams} \\ w_1 &= 0.8 \\ w_2 &= 0.2 \\ w_3 &= 3 \\ b &= 65 \end{aligned}$$

For this application the quantity $\sum_j w_j x_j$ in Equation A.1 represents the student's expected score. If this predicted score is greater than the traditional pass/fail cutoff of $b = 65$ points, we expect the student to pass the exam. Otherwise, the student is predicted to fail the last exam.

Motivation for Activation Functions

Although this model is useful for predicting whether a given student will pass or fail the second exam, it is difficult to adjust the above weights to improve our model over time (a process called "backpropagation") because we have no means of knowing how far off our predictions were. If for an incorrect prediction $\sum_j w_j x_j \approx b$, then our model was not wildly inaccurate and the model weights should perhaps be slightly adjusted. On the other hand, if $\sum_j w_j x_j \gg b$, then our model was way off the mark and our model's weights may need significant adjustment. In order to avoid the discontinuous unit step function used above, it is typically replaced with a non-linear but continuous "activation function" in modern implementations. Three of the most common activation functions are described below and then shown in Figure A.3:

sigmoid- named after its S-shaped form, it maps a real number to the range between 0 and 1 and visually appears to be a smoother unit step function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

tanh- the hyperbolic tangent function, it maps a real number to the range between -1 and 1:

$$\tanh(x) = 2\sigma(x) - 1$$

ReLU- the rectified linear unit function, it performs minimum thresholding at zero:

$$f(x) = \max(0, x)$$

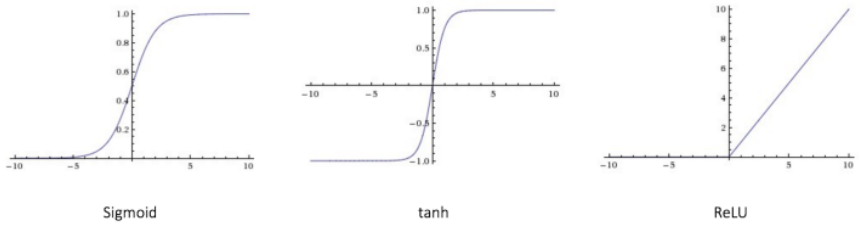

Figure A.3: Various Activation Functions [33]

A.1.3 Networks of Neurons

For most applications, using a single neuron is insufficient to model the nonlinear complexities of the given problem. In order to do so, it becomes necessary to combine numerous neurons together in a network of several layers, where the inputs to each layer are the outputs of neurons at the previous layer. A basic fully-connected neural network architecture is shown below in Figure A.4. This network is four layers deep, with two internal "hidden layers" which comprise the bulk of the classifier.

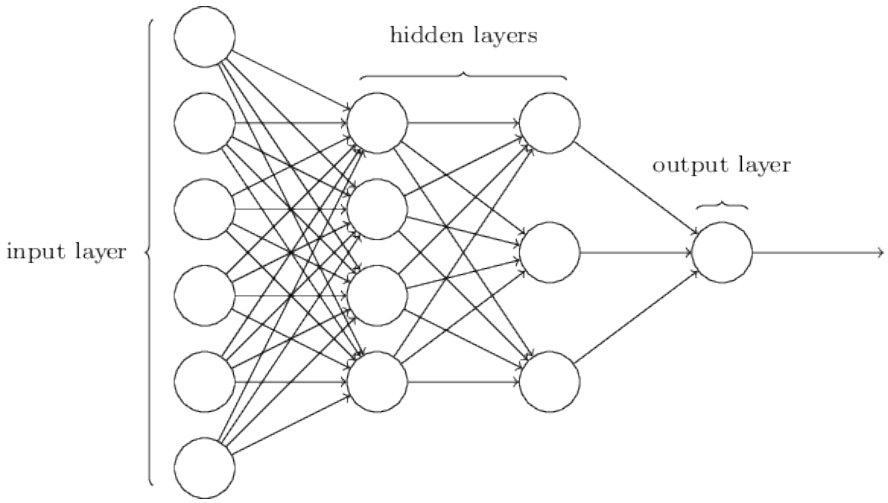

Figure A.4: Neural Network Architecture [32]

A.2 Convolutional Neural Networks

A.2.1 Overview

Also known as CNNs or ConvNets, Convolutional Neural Networks are a biologically-inspired class of deep learning models which are specifically designed to classify raw image data. As discussed in Section A.1, traditional neural networks make predictions given a vector of independent input variables. Since image pixels are not independent of one another and there are strong local patterns, artificial neural networks have historically not performed well in classifying images. Rather than transforming a vector using a one-dimensional array of neurons in each of several hidden layers, a convolutional neural network performs some differentiable function (which may or may not have input parameters) on a three-dimensional volume at each layer to generate another three-dimensional volume.

As can be seen below in Figure A.5, a CNN gradually transforms the image data over the course of many layers, slowly flattening a large 2-dimensional matrix into a 1-dimensional vector. The input volume's length along the third dimension is steadily increased by performing convolution with a greater number of kernels, and its length

along the first two dimensions is reduced exponentially by down-sampling. This can again be seen in Figure A.5.

CNNs will gradually transform an input image to a single classification result. The convolutional layers allow the classifier to identify two-dimensional spatial patterns without requiring fully connected layers which introduce too many parameters. Small kernels and early convolutional layers will identify localized patterns whereas larger kernels and later layers will recognize more general trends.

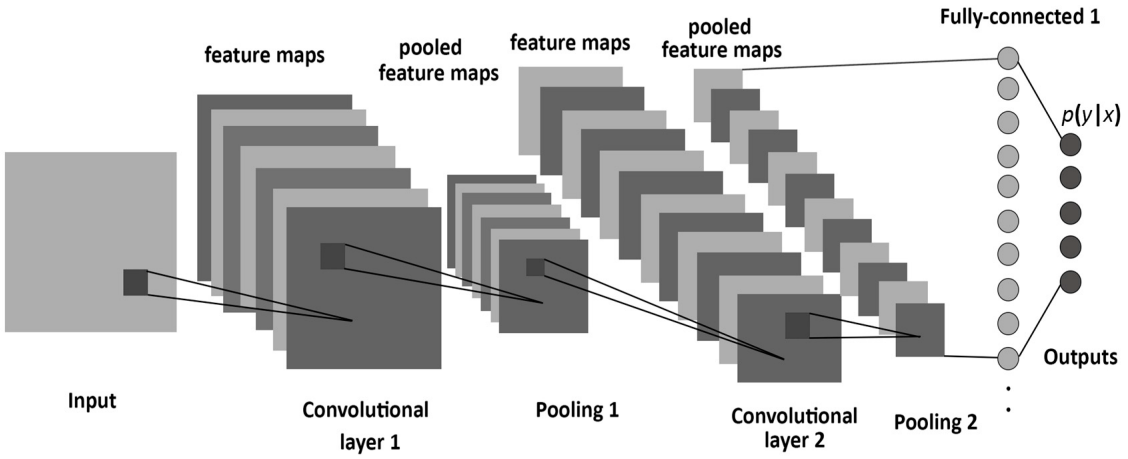

Figure A.5: CNN Architecture [35]

A.2.2 Layers

There are many different types of ConvNet layers, each designed to serve a specific function. What follows is a brief summarization of the most commonly used layers.

Input Layer

The input layer holds the raw pixel values of an image. In the case of a grayscale image, the dimensions of this layer are $w \times h$. This layer is analogous to the input layer of an ANN, pictured in Figure A.4.

Convolutional Layer

A convolutional layer computes the convolution of the input volume with one or more smaller kernels of the same size. If a convolutional layer immediately follows the input layer of a $w \times h$ grayscale image and 12 kernels are used, the output volume will be $w \times h \times 12$ (assuming padding is used to retain image shape). Figure A.6 shows an example of the convolution operation applied to image I with kernel K. Note that in this example, no padding is used and the resulting image $I * K$ decreases in size by $2k$, where k is the kernel width.

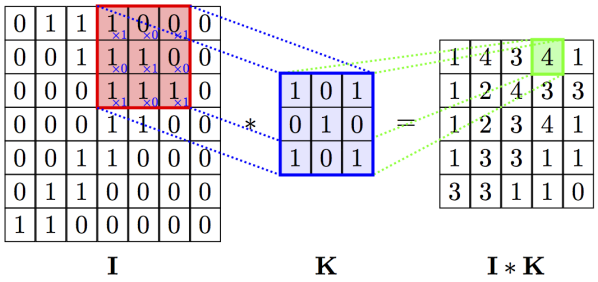

Figure A.6: Convolution Example [34]

If kernel K is size $k \times k$, then the radius r of kernel K is defined by $r = \lfloor k/2 \rfloor$. A more formal mathematical

definition of convolving image I with kernel K of radius r would then be:

$$I[y,x] \circledast K = \sum_{j=0}^{2r} \sum_{i=0}^{2r} I[y-r+i, x-r+j] * K[j,i]$$

ReLU Layer

A rectified linear unit layer will perform the ReLU activation function (shown in Figure A.3c) element-wise on the input. This leaves the dimensions of the input volume unchanged.

Pooling Layer

A pooling layer will perform down-sampling along the first two dimensions of the input, resulting in a smaller output volume. Down-sampling is most commonly performed with the `max()` operation, where the maximum pixel is selected within each block of $s \times s$ pixels (where s is called the "stride length"), giving rise to maximum pooling. Figure A.7 shows an example of maximum pooling with stride length of 2, which is most commonly used in practice.

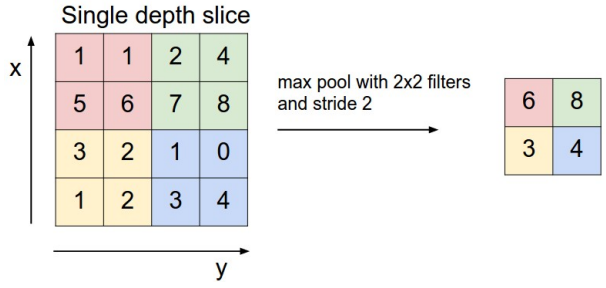

Figure A.7: Maximum Pooling [32]

Fully-Connected Layer

Once an image has been sufficiently down-sampled, the input volume is flattened so that it becomes one-dimensional and a final fully-connected layer is used to make a prediction. This fully-connected layer is the same as a hidden or output layer of an ordinary neural network, shown in Figure A.4.

A.3 Image Processing

A.3.1 Morphological Filtering

For the purposes of this discussion, we consider only morphological operations on binary black and white images.

Erosion & Dilation

The two most fundamental morphological operations are erosion and dilation. Each operation is performed on a binary image using a "kernel" or "structuring element" K. In the example shown in Figures A.8 and A.9,

$$K = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} =$$

(kernel not shown to scale). Notice that the central pixel in kernel K is white, as are most pixels immediately surrounding it. During the dilation operation, the result of which is shown in Figure A.8c, the kernel K is laid as a

template over the original image. If the center pixel of kernel K is placed over a white pixel, then any black pixels in the original image which are located under a white pixel in the kernel are converted to white. During erosion the opposite occurs, and white pixels in the original image which are located under a white pixel in the kernel when the central kernel pixel is black are recolored black, the result of which is shown in Figure A.8b. Individually, these two operations have the effect of either dilating or eroding the boundary of white pixels in the image, and are thus aptly named.

Figure A.8: Basic Morphological Operations: (a) Original Image (b) Erosion (c) Dilation [36]

Opening & Closing

The "opening" and "closing" operations are equivalent to an erosion followed by a dilation, and a dilation followed by an erosion, respectively. At first glance, it appears that eroding and then dilating an image should result in the original image. However, this is not quite the case. If there is an isolated patch of white pixels which is smaller than the kernel used to perform the erosion and dilation, then this patch will be fully recolored black during the erosion operation and will not reappear after dilating the image. An example of this can be seen below, in Figure A.9a. In fact, structuring elements of various sizes are often used specifically to remove unwanted patterns from an image. This small 7×7 kernel is well-suited to removing the salt-and-pepper noise of Figure $A.9$. If we instead wanted to remove only thin vertical lines with widths of approximately 5 pixels, we would use something like a 1×8 kernel, which would remove from the image elements with widths of less than 8 pixels. Of course, the conjugate is true for the closing operation, which can be used to effectively remove black pixels from a white background. This operation can be seen in Figure A.9b, again applied with kernel K.

Figure A.9: Two-Step Morphological Operations: (a) Opening (b) Closing [36]

Hit-and-Miss

The "hit-and-miss" transformation is used to find patterns in binary images, where it finds regions of pixels which match kernel K_1 while not matching kernel K_2 [37]. The hit-and-miss transformation thus comprises of three steps:

1. Eroding image A with structuring element K_1.
2. Eroding the complement of image A with structuring element K_2.

3. Performing a logical AND operation on the two previous results.

The construction of a hit-and-miss kernel can be seen in Figure A.10. This filter will act as a template which is laid over each position within the original image. In this example, the result will be 1 only if the image pixel below the central hit-and-miss kernel pixel is 0 and all four adjacent pixels are 1. The diagonally-adjacent (kernel corner) pixels have no effect. An example application of this kernel is shown in Figure A.11.

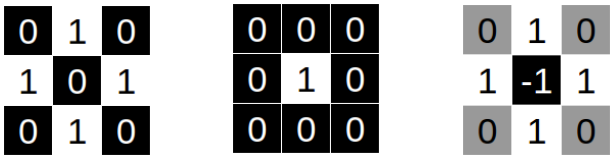

Figure A.10: (a) "Hit" kernel (b) "Miss" kernel (c) "Hit-and-Miss" kernel

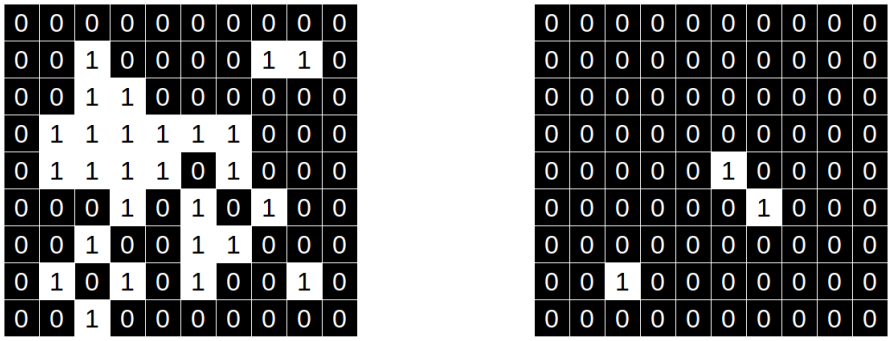

Figure A.11: "Hit-and-Miss" Kernel Example

A.3.2 Optical Flow

Optical flow is the pattern of apparent motion of image objects between consecutive frames in a video. Dense optical flow is a computed uniform 2D vector field corresponding to the apparent motion of each image pixel. Figure A.12 shows an example where optical flow has been computed using the Lucas-Kanade algorithm on a video of horses running along the shore.

Figure A.12: Dense Optical Flow Example [38]

Bibliography

[1] https://medium.com/nanonets/nanonets-how-to-use-deep-learning-when-you-have-limited-data-f68c0b512cab

[2] Wang, Heng, and Cordelia Schmid. "Action recognition with improved trajectories." *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2013.

[3] Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." *IEEE Transactions on Systems and Cybernetics*, 1979.

[4] LeCun, Yann. "The MNIST database of handwritten digits." http://yann. lecun. com/exdb/mnist/ (1998).

[5] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah. " UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild." CRCV-TR-12-01, 2012.

[6] C. Gao, Y. Du, J. Liu, J. Lv, L. Yang, D. Meng, and A. Hauptmann. "Infar dataset: Infrared action recognition at different times." *Neurocomputing, 212:36âĂŞ47*, 2016.

[7] Skala, Karolj, et al. "4D thermal imaging system for medical applications." Periodicum biologorum 113.4 (2011): 407-416.

[8] FLIR Systems. "Vue Pro and Vue Pro R User Guide." Document Number: 436-0013-10, Version 110. July 2016.

[9] T. Dunn, S. Banerjee, and N. K. Banerjee. "User-Independent Detection of Swipe Pressure using a Thermal Camera for Natural Surface Interaction." *20th International Workshop on Multimedia Signal Processing (MMSP).* 2018.

[10] Pierre Wellner. "Interacting with paper on the DigitalDesk." Communications of the ACM, vol. 36, no. 7, pp. 87âĂŞ96, 1993.

[11] Kenji Oka, Yoichi Sato, and Hideki Koike. "Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems." *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition.* 2002.

[12] Daisuke Iwai and Kosuke Sato. "Heat sensation in image creation with thermal vision." *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology.* 2005.

[13] Andrew Wilson. "Playanywhere: A compact interactive tabletop projection-vision system." *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology.* 2005.

[14] Pranav Mistry and Pattie Maes. "Sixthsense: a wearable gestural interface.âĂİ *ACM SIGGRAPH ASIA 2009 Sketches.* 2009.

[15] Eric Larson et al. "Heatwave: thermal imaging for surface user interaction." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 2011.

[16] Elliot Saba, Eric Larson, and Shwetak Patel. "Dante vision: In-air and touch gesture sensing for natural surface interaction with combined depth and thermal cameras." *2012 IEEE International Conference on Emerging Signal Processing Applications (ESPA).* 2012.

[17] Daniel Kurz. "Thermal touch: Thermography-enabled everywhere touch interfaces for mobile augmented reality applications." *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR).* 2014.

[18] Karri Palovuori and Ismo Rakkolainen. "The heat is on: thermal input for immaterial interaction." *Proceedings of the 19th International Academic Mindtrek Conference.* 2015.

[19] LeCun, Yann, et al. "Object recognition with gradient-based learning." *Shape, contour and grouping in computer vision.* Springer, Berlin, Heidelberg, 1999.

[20] Ji, Shuiwang, et al. "3D convolutional neural networks for human action recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence 35.1*, 2013.

[21] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2014.

[22] Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." *Advances in Neural Information Processing Systems.* 2014.

[23] Tran, Du, et al. "Learning spatiotemporal features with 3d convolutional networks." *Proceedings of the IEEE Conference on Computer Vision.* 2015.

[24] Yue-Hei Ng, Joe, et al. "Beyond short snippets: Deep networks for video classification." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015.

[25] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[26] Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman. "Convolutional two-stream network fusion for video action recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016.

[27] Jiang, Zhuolin, Viktor Rozgic, and Sancar Adali. "Learning spatiotemporal features for infrared action recognition with 3d convolutional neural networks." *Proceedings of the IEEE Computer Vision and Pattern Recognition Workshops.* 2017.

[28] Carreira, Joao, and Andrew Zisserman. "Quo vadis, action recognition? A new model and the kinetics dataset." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017.

[29] Diba, Ali, et al. "Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification." *arXiv preprint arXiv:1711.08200.* 2017.

[30] Varol, Guel, Ivan Laptev, and Cordelia Schmid. "Long-term temporal convolutions for action recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2018.

[31] http://ib.bioninja.com.au/standard-level/topic-6-human-physiology/65-neurons-and-synapses/neurons.html

[32] http://cs231n.github.io/convolutional-networks/

[33] https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

[34] https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution

[35] S. Albelwi, A. Mahmood. "A Framework for Designing the Architectures of Deep Convolutional Neural Networks." *Entropy.* 2017.

[36] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morpholog

[37] https://docs.opencv.org/3.4/db/d06/tutorial_hitOrMiss.html

[38] http://cs.brown.edu/courses/csci1290/2011/results/final/psastras/

[39] http://alexlenail.me/NN-SVG/LeNet.html